main.c

Share Run

```c
#include <stdio.h>
#include <stdlib.h>
typedef struct Node {
    int data;
    struct Node* next;
} Node;
int countNodes(Node* head) {
    int count = 0;
    Node* temp = head;
    while (temp != NULL) {
        count++;
        temp = temp->next;
    }
    return count;
}
Node* createNode(int data) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    if (!newNode) {
        printf("Memory error\n");
        return NULL;
    }
    newNode->data = data;
    newNode->next = NULL;
    return newNode;
}

```

Output Clear

```
/tmp/wQY6QGCy7W.o
Linked List: 1->2->3->5->8->NULL
Number of nodes: 5


=== Code Execution Successful ===
```

main.c

Share Run

```c
#include <stdio.h>
void insertionSort(int arr[], int n) {
    int i, key, j;
    for (i = 1; i < n; i++) {
        key = arr[i];
        j = i - 1;
        while (j >= 0 && arr[j] > key) {
            arr[j + 1] = arr[j];
            j = j - 1;
        }
        arr[j + 1] = key;
        printf("Iteration %d: ", i);
        for (int k = 0; k < n; k++) {
            printf("%d ", arr[k]);
        }
        printf("\n");
    }
}
int main() {
    int arr[] = {98, 23, 45, 14, 6, 67, 33, 42};
    int n = sizeof(arr) / sizeof(arr[0]);

    printf("Original array: ");
    for (int i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
```

Output Clear

```
/tmp/znYfYC736w.o
Original array: 98 23 45 14 6 67 33 42
Iteration 1: 23 98 45 14 6 67 33 42
Iteration 2: 23 45 98 14 6 67 33 42
Iteration 3: 14 23 45 98 6 67 33 42
Iteration 4: 6 14 23 45 98 67 33 42
Iteration 5: 6 14 23 45 67 98 33 42
Iteration 6: 6 14 23 33 45 67 98 42
Iteration 7: 6 14 23 33 42 45 67 98
Sorted array: 6 14 23 33 42 45 67 98


=== Code Execution Successful ===
```

main.c

Share Run

```c
#include <stdio.h>
#include <stdbool.h>
#define MAX 100005
int adj[MAX];
bool visited[MAX];
int dist[MAX];
void bfs(int start, int n) {
    int queue[MAX], front = 0, rear = 0;
    queue[rear++] = start;
    visited[start] = true;
    dist[start] = 0;

    while (front < rear) {
        int node = queue[front++];
        for (int i = 0; i < n; i++) {
            if (adj[node * n + i] == 1 && !visited[i]) {
                queue[rear++] = i;
                visited[i] = true;
                dist[i] = dist[node] + 1;
            }
        }
    }
}

int main() {
    int n, m, u, v;
```

Output

Clear

```
1
2
3
4
5
Minimum number of edges between 4 and 5 is 0


=== Code Execution Successful ===
```

main.c

Share Run

Output Clear

```c
#include <stdio.h>
#include <stdlib.h>

typedef struct TreeNode {
    int val;
    struct TreeNode* left;
    struct TreeNode* right;
} TreeNode;

void inorderTraversal(TreeNode* root, int* count, int k, int* result) {
    if (root == NULL) {
        return;
    }

    inorderTraversal(root->left, count, k, result);

    (*count)++;

    if (*count == k) {
        *result = root->val;
        return;
    }

    inorderTraversal(root->right, count, k, result);
}
```

```
/tmp/pYqWCXceBQ.o
Kth smallest element is 4
Kth smallest element is 7


=== Code Execution Successful ===
```

main.c

Share Run

```c
#include <stdio.h>
#include <string.h>

void charFrequency(char* s) {
    int freq[256] = {0};

    for (int i = 0; i < strlen(s); i++) {
        freq[(int)s[i]]++;
    }

    for (int i = 0; i < 256; i++) {
        if (freq[i] > 0) {
            printf("%c->%d, ", (char)i, freq[i]);
        }
    }
    printf("\n");
}

int main() {
    char s[] = "tree";
    charFrequency(s);

    return 0;
}
```

Output Clear

```
/tmp/dKn8nuB1Jn.o
e->2, r->1, t->1,


=== Code Execution Successful ===
```

main.c

Share Run

```c
#include <stdbool.h>
#include <stdio.h>
#include <stdlib.h>

typedef struct ListNode {
    int val;
    struct ListNode* next;
} ListNode;

ListNode* createListNode(int val) {
    ListNode* node = (ListNode*)malloc(sizeof(ListNode));
    node->val = val;
    node->next = NULL;
    return node;
}

void printList(ListNode* head) {
    while (head != NULL) {
        printf("%d ", head->val);
        head = head->next;
    }
    printf("\n");
}

bool isPalindrome(ListNode* head) {
    if (head == NULL || head->next == NULL) {
```

Output Clear

```
/tmp/fJLqOunx4y.o
1 2 3 2 1
true
1 2 3


=== Code Execution Successful ===n
```

main.c

Share Run

```c
#include <stdio.h>
int fibonacci(int n) {
    if (n == 0 || n == 1) {
        return n;
    } else {
        return fibonacci(n - 1) + fibonacci(n - 2);
    }
}
void printFibonacciSeries(int n) {
    printf("Fibonacci series: ");
    for (int i = 0; i < n; i++) {
        printf("%d, ", fibonacci(i));
    }
    printf("\n");
}
int sumFibonacciSeries(int n) {
    int sum = 0;
    for (int i = 0; i < n; i++) {
        sum += fibonacci(i);
    }
    return sum;
}

int main() {
    int n;
    printf("Enter the number of terms: ");
```

Output Clear

```
/tmp/6pO5UbXrzV.o
Enter the number of terms: 5
Fibonacci series: 0, 1, 1, 2, 3,
Sum of the Fibonacci series: 7


=== Code Execution Successful ===
```

main.c

Share Run

```c
#include <stdio.h>
int binarySearch(int arr[], int x, int low, int high) {
    if (high >= low) {
        int mid = low + (high - low) / 2;
        if (arr[mid] == x) {
            return mid;
        }
        if (arr[mid] > x) {
            return binarySearch(arr, x, low, mid - 1);
        }
        return binarySearch(arr, x, mid + 1, high);
    }
    return -1;
}

int main() {
    int arr[] = {1, 5, 6, 7, 9, 10};
    int x = 6;
    int n = sizeof(arr) / sizeof(arr[0]);

    int result = binarySearch(arr, x, 0, n - 1);

    if (result == -1) {
        printf("Element not found\n");
    } else {
        printf("Element found at location %d\n", result);
```

Output Clear

```
/tmp/u1f0NS8DQf.o
Element found at location 2


=== Code Execution Successful ===
```

main.c

Share Run

```c
#include <stdio.h>
#include <stdlib.h>
typedef struct Node {
    int data;
    struct Node* left;
    struct Node* right;
} Node;
Node* createNode(int data) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    if (!newNode) {
        printf("Memory error\n");
        return NULL;
    }
    newNode->data = data;
    newNode->left = newNode->right = NULL;
    return newNode;
}
void inorderTraversal(Node* root) {
    if (root) {
        inorderTraversal(root->left);
        printf("%d ", root->data);
        inorderTraversal(root->right);
    }
}
void postorderTraversal(Node* root) {
    if (root) {
```

Output Clear

```
/tmp/8WVHHBpbxr.o
Inorder traversal: 9 3 15 20 7
Postorder traversal: 9 15 7 20 3


=== Code Execution Successful ===
```

main.c

Share Run

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define MAX 256
typedef struct {
    char c;
    int freq;
} CharFreq;

int compare(const void *a, const void *b) {
    CharFreq *cf1 = (CharFreq *)a;
    CharFreq *cf2 = (CharFreq *)b;
    if (cf1->freq != cf2->freq) {
        return cf2->freq - cf1->freq;
    } else {
        return cf1->c - cf2->c;
    }
}
void sort_and_find_repeated(char *s) {
    int len = strlen(s);
    CharFreq cf[MAX];
    int count = 0;
    for (int i = 0; i < len; i++) {
        int found = 0;
        for (int j = 0; j < count; j++) {
            if (cf[j].c == s[i]) {
```

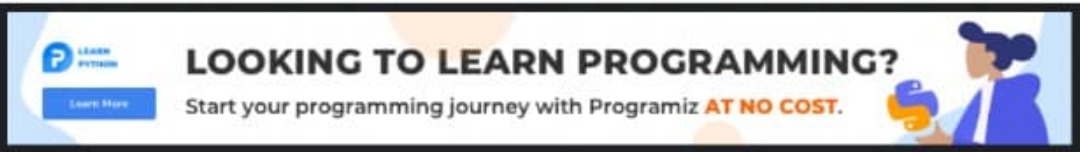
Output Clear

```
/tmp/XCmA3xNYet.o
Sorted string: eert
Starting indices of repeated characters: 2
Sorted string: kkj
Starting indices of repeated characters: 1
Sorted string: aaaccc
Starting indices of repeated characters: 3 1
Sorted string: bbAa
Starting indices of repeated characters: 2


=== Code Execution Successful ===
```

Online C Compiler - Programiz | You are given with an array arr

programiz.com/c-programming/online-compiler/

Programiz
C Online Compiler



Programiz PRO >

main.c | Share | Run

```c
#include <stdio.h>
#include <stdlib.h>
typedef struct TreeNode {
    int val;
    struct TreeNode* left;
    struct TreeNode* right;
} TreeNode;
TreeNode* createNode(int val) {
    TreeNode* newNode = (TreeNode*)malloc(sizeof(TreeNode));
    if (!newNode) {
        printf("Memory error\n");
        return NULL;
    }
    newNode->val = val;
    newNode->left = NULL;
    newNode->right = NULL;
    return newNode;
}
TreeNode* buildTree(int* preorder, int* inorder, int preorderStart, int
    preorderEnd, int inorderStart, int inorderEnd) {
    if (preorderStart > preorderEnd || inorderStart > inorderEnd) {
        return NULL;
    }
    TreeNode* root = createNode(preorder[preorderStart]);
    int inorderIndex;
    for (inorderIndex = inorderStart; inorderIndex <= inorderEnd; inorderIndex
```

Output | Clear

```
/tmp/bbCCcQXq4h.o
Binary Tree:
    7
  20
    15
3
  9


=== Code Execution Successful ===
```

30°C Haze | Search | ENG US | 09:41 26-08-2024

Online C Compiler - Programiz | You are given with an array arr

programiz.com/c-programming/online-compiler/

Programiz
C Online Compiler

Programiz PRO >

main.c | Share | Run | Output | Clear

```c
#include <stdio.h>
int findSmallestPositiveMissing(int arr[], int n) {
    int i, j, temp;
    for (i = 0, j = 0; i < n; i++) {
        if (arr[i] > 0) {
            temp = arr[i];
            arr[i] = arr[j];
            arr[j] = temp;
            j++;
        }
    } index x
    for (i = 0; i < j; i++) {
        if (abs(arr[i]) - 1 < j && arr[abs(arr[i]) - 1] > 0) {
            arr[abs(arr[i]) - 1] = -arr[abs(arr[i]) - 1];
        }
    }
    for (i = 0; i < j; i++) {
        if (arr[i] > 0) {
            return i + 1;
        }
    }
    return j + 1;
}
int main() {
    int arr[] = {2, 3, 7, 6, 8, -1, -10, 15};
    int n = sizeof(arr) / sizeof(arr[0]);
```

```
/tmp/pBDaEfyO2Q.o
Linked List: 1->2->3->5->8->NULL
Number of nodes: 5


=== Code Execution Successful ===
```

30°C Haze | Search | ENG US | 09:39 26-08-2024

Online C Compiler - Programiz | You are given with an array arr

programiz.com/c-programming/online-compiler/

Programiz
C Online Compiler

Programiz PRO >

main.c | Share | Run | Output | Clear

```c
#include <stdio.h>
void bubbleSort(int arr[], int n) {
    int i, j, temp;
    for (i = 0; i < n - 1; i++) {
        for (j = 0; j < n - i - 1; j++) {
            if (arr[j] < arr[j + 1]) {
                temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
            }
        }
    }
}
void printArray(int arr[], int n) {
    int i;
    for (i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");
}
int main() {
    int arr[] = {4, 7, 9, 1, 2};
    int n = sizeof(arr) / sizeof(arr[0]);
    printf("Original array: ");
    printArray(arr, n);
    bubbleSort(arr, n);
```

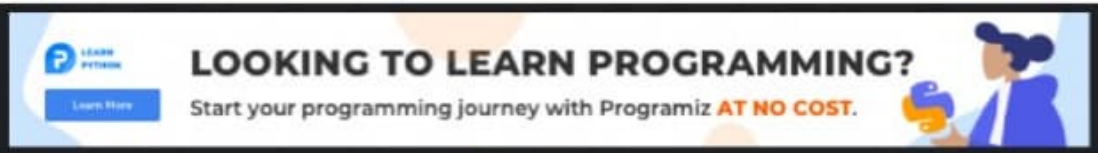
```
/tmp/oNIi17BChJ.o
Original array: 4 7 9 1 2
Sorted array in descending order: 9 7 4 2 1


=== Code Execution Successful ===
```

30°C Haze | Search | ENG US | 09:44 26-08-2024

Online C Compiler - Programiz | You are given with an array arr

programiz.com/c-programming/online-compiler/

Programiz
C Online Compiler



Programiz PRO ›

main.c | Share | Run | Output | Clear

```c
#include <stdio.h>
#include <stdlib.h>
typedef struct Node {
    int data;
    struct Node* next;
} Node;
void insertNode(Node** head, int data) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    newNode->data = data;
    newNode->next = NULL;
    if (*head == NULL) {
        *head = newNode;
    } else {
        Node* temp = *head;
        while (temp->next != NULL) {
            temp = temp->next;
        }
        temp->next = newNode;
    }
}
void printOddNumbers(Node* head) {
    while (head != NULL) {
        if (head->data % 2 != 0) {
            printf("%d ", head->data);
        }
        head = head->next;
```

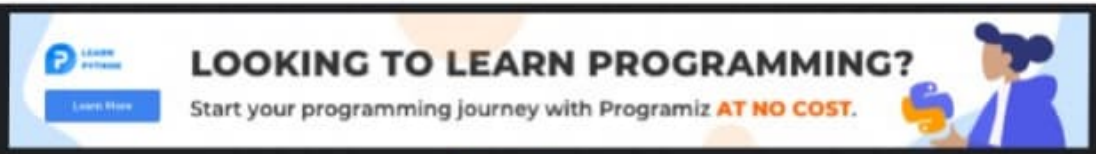
```
/tmp/Xwn7zZZcfz.o
Odd numbers in the list: 1 3 7


=== Code Execution Successful ===
```

30°C Haze | Search | ENG US | 09:46 26-08-2024

Online C Compiler - Programiz | You are given with an array arr

programiz.com/c-programming/online-compiler/

Programiz
C Online Compiler



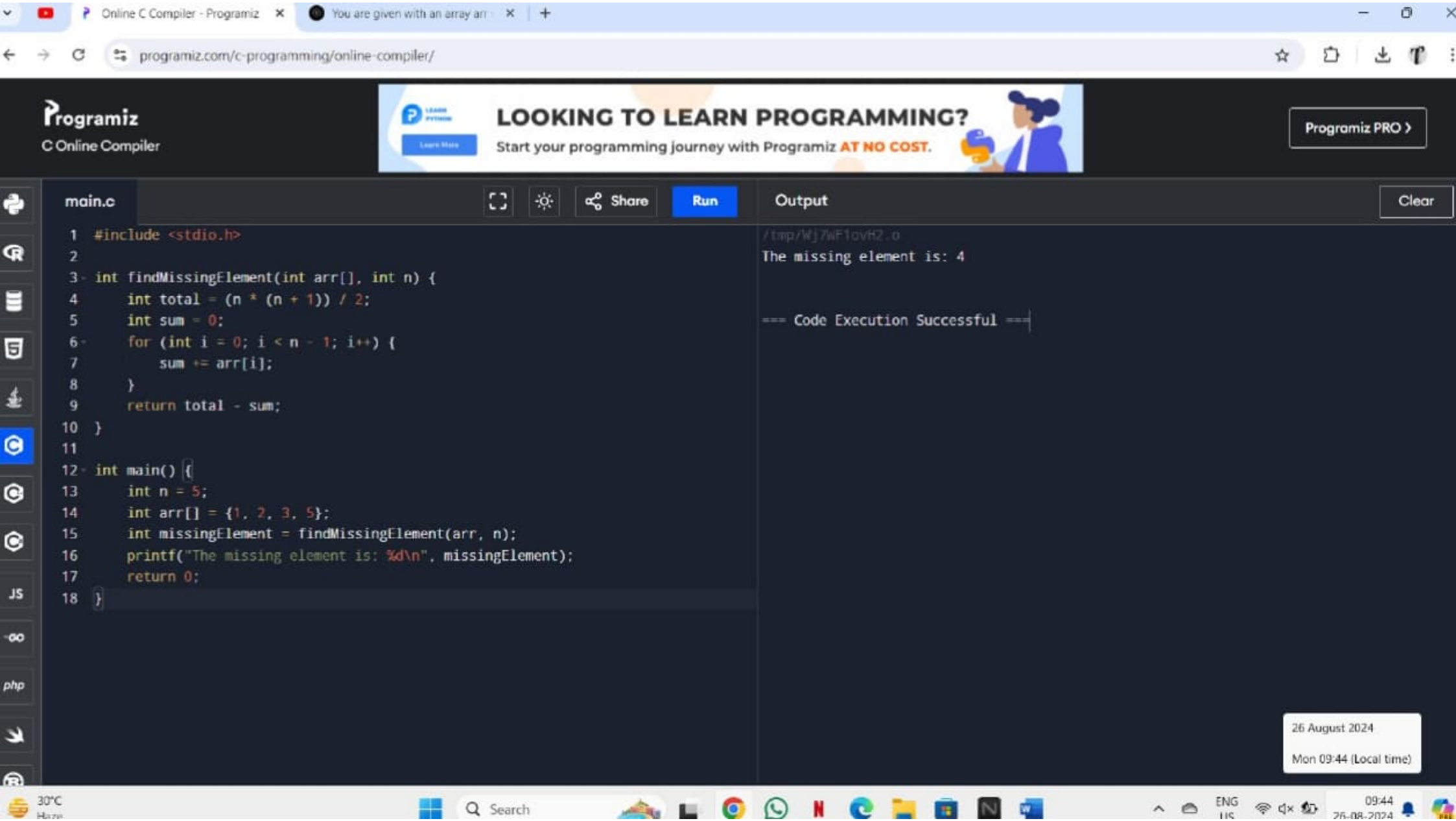
Programiz PRO >

main.c | Share | Run | Output | Clear

```c
#include <stdio.h>
#include <stdlib.h>
typedef struct Node {
    int data;
    struct Node* next;
} Node;
typedef struct Queue {
    Node* front;
    Node* rear;
} Queue;
Node* createNode(int data) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    newNode->data = data;
    newNode->next = NULL;
    return newNode;
}
Queue* createQueue() {
    Queue* q = (Queue*)malloc(sizeof(Queue));
    q->front = NULL;
    q->rear = NULL;
    return q;
}
void enqueue(Queue* q, int data) {
    Node* newNode = createNode(data);
    if (q->rear == NULL) {
        q->front = newNode;
```

```
/tmp/kQKA3wqtVY.o
Initial queue: 12 34 56 78
After insertion of 60: 12 34 56 78 60
After deletion of 12: 34 56 78 60


=== Code Execution Successful ===
```

30°C Haze | Search | ENG US | 09:47 26-08-2024


Online C Compiler - Programiz
You are given with an array arr
programiz.com/c-programming/online-compiler/
Programiz
C Online Compiler
LOOKING TO LEARN PROGRAMMING?
Start your programming journey with Programiz AT NO COST.
Programiz PRO
main.c
Share
Run
Output
Clear
#include <stdio.h>
int findMissingElement(int arr[], int n) {
int total = (n * (n + 1)) / 2;
int sum = 0;
for (int i = 0; i < n - 1; i++) {
sum += arr[i];
}
return total - sum;
}
int main() {
int n = 5;
int arr[] = {1, 2, 3, 5};
int missingElement = findMissingElement(arr, n);
printf("The missing element is: %d\n", missingElement);
return 0;
}
/tmp/Wj7wF1ovH2.o
The missing element is: 4
=== Code Execution Successful ===
26 August 2024
Mon 09:44 (Local time)
30°C
Haze
Search
ENG
US
09:44
26-08-2024

Online C Compiler - Programiz | You are given with an array an

programiz.com/c-programming/online-compiler/

Programiz
C Online Compiler

main.c | Share | Run | Output | Clear

```c
#include <stdio.h>
#include <stdlib.h>
typedef struct Node {
    int data;
    struct Node* next;
} Node;
Node* createNode(int data) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    if (!newNode) {
        printf("Memory error\n");
        return NULL;
    }
    newNode->data = data;
    newNode->next = NULL;
    return newNode;
}
void insertNode(Node** head, int data) {
    Node* newNode = createNode(data);
    if (*head == NULL) {
        *head = newNode;
        return;
    }
    Node* lastNode = *head;
    while (lastNode->next) {
        lastNode = lastNode->next;
    }
```

```
/tmp/GQ21SPjYuQ.o
6->7->8->9->NULL


=== Code Execution Successful ===
```

30°C Haze | Search | ENG US | 09:43 26-08-2024

Online C Compiler - Programiz | You are given with an array an

programiz.com/c-programming/online-compiler/

Programiz
C Online Compiler

main.c | Share | Run | Output | Clear

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int isValid(char *s) {
    int stack[1000];
    int top = -1;
    char mapping[256];
    mapping[')'] = '(';
    mapping['}'] = '{';
    mapping[']'] = '[';
    for (int i = 0; i < strlen(s); i++) {
        if (s[i] == '(' || s[i] == '{' || s[i] == '[') {
            stack[++top] = s[i];
        } else if (s[i] == ')' || s[i] == '}' || s[i] == ']') {
            if (top == -1 || stack[top] != mapping[s[i]]) {
                return 0;
            }
            top--;
        }
    }
    return top == -1;
}
int main() {
    char s[] = "()";
    if (isValid(s)) {
        printf("true\n");
```

```
/tmp/YzreughnnP.o
true


=== Code Execution Successful ===
```

30°C Haze | Search | ENG US | 09:48 26-08-2024

Online C Compiler - Programiz | You are given with an array arr

programiz.com/c-programming/online-compiler/

Programiz
C Online Compiler

main.c

Share | Run

```c
#include <stdio.h>
void printFibonacci(int n) {
    int t1 = 0, t2 = 1, nextTerm = 0, sum = 1;
    printf("Fibonacci series: ");
    printf("%d, %d, ", t1, t2);
    for (int i = 3; i <= n; i++) {
        nextTerm = t1 + t2;
        t1 = t2;
        t2 = nextTerm;
        sum += nextTerm;
        printf("%d, ", nextTerm);
    }
    printf("\nSum of the series: %d\n", sum);
}
int main() {
    int n;
    printf("Enter the number of terms: ");
    scanf("%d", &n);
    printFibonacci(n);
    return 0;
}
```

Output | Clear

```
/tmp/RAKxGtOAWV.o
Enter the number of terms: 10
Fibonacci series: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34,
Sum of the series: 88


=== Code Execution Successful ===
```

30°C Haze | Search | ENG US | 09:49 26-08-2024

Online C Compiler - Programiz | You are given with an array an

programiz.com/c-programming/online-compiler/

Programiz
C Online Compiler

Programiz PRO >

main.c | Share | Run

```c
#include <stdio.h>
#include <string.h>
int strStr(char *haystack, char *needle) {
    int haystackLen = strlen(haystack);
    int needleLen = strlen(needle);
    if (needleLen == 0) {
        return 0;
    }
    for (int i = 0; i <= haystackLen - needleLen; i++) {
        int j;
        for (j = 0; j < needleLen; j++) {
            if (haystack[i + j] != needle[j]) {
                break;
            }
        }
        if (j == needleLen) {
            return i;
        }
    }
    return -1;
}
int main() {
    char haystack[] = "sadbutsad";
    char needle[] = "sad";
    int index = strStr(haystack, needle);
    if (index != -1) {
```

Output | Clear

```
/tmp/JC5MaCrSOV.o
The first occurrence of 'sad' in 'sadbutsad' is at index 0.


=== Code Execution Successful ===
```

30°C Haze | Search | ENG US | 09:50 26-08-2024